PAT-NO: JP411238004A

DOCUMENT-IDENTIFIER: JP 11238004 A

TITLE: SYSTEM SIMULATOR

PUBN-DATE: August 31, 1999

INVENTOR-INFORMATION:

NAME	COUNTRY
ANDO, MITSUO	N/A
AKIYOSHI, KUNIHIRO	N/A

ASSIGNEE-INFORMATION:

NAME	COUNTRY
RICOH CO LTD	N/A

APPL-NO: JP10055970

APPL-DATE: February 20, 1998

INT-CL (IPC): G06F011/28, G06F011/28 , G06F009/455 , G06F009/46

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a system simulator constructing a simulation environment on a frame work.

SOLUTION: A system simulator 1 is provided with a CPU simulator 3 simulating target CPU, an input/output I/O simulator 4 simulating a target input/ output device and a frame work 5 connecting the CPU simulator 3 and the input/output I/O simulator 4 with object communication are provided. A simulation environment which can execute a target program 2 by a host CPU system can be constructed on the frame work 5 directing the object by

transferring respective
instruction codes with object communication. The environment of the
development of firmware and the development of the CPU system can be improved,
firmware and the CPU system can easily and speedily be developed. Then, speedy
development can be executed in the CPU system of the different input/output
device with the free combination of the objects.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平11－238004

(43)公開日　平成11年(1999)８月31日

(51)Int.Cl.⁶	識別記号	ＦＩ	
Ｇ０６Ｆ　11/28	３４０	Ｇ０６Ｆ　11/28	３４０Ｃ
			Ｊ
9/455		9/46	３６０Ｆ
9/46	３６０	9/44	３１０Ｄ

審査請求　未請求　請求項の数４　ＦＤ　（全　８　頁）

(21)出願番号　特願平10－55970

(22)出願日　平成10年(1998)２月20日

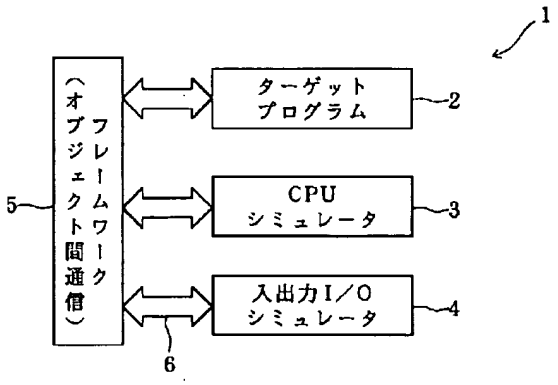
(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号

(72)発明者　安藤　光男
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(72)発明者　秋吉　邦洋
東京都大田区中馬込１丁目３番６号　株式会社リコー内

(54)【発明の名称】　システムシミュレータ

(57)【要約】
【課題】本発明はフレームワーク上でシミュレーション環境を構築するシステムシミュレータを提供する。
【解決手段】システムシミュレータ１は、ターゲットＣＰＵを模擬するＣＰＵシミュレータ３と、ターゲット入出力装置を模擬する入出力Ｉ／Ｏシミュレータ４と、ＣＰＵシミュレータ３及び入出力Ｉ／Ｏシミュレータ４をオブジェクト通信で連結するフレームワーク５と、を備え、ターゲットプログラム２をホストＣＰＵシステムで実行可能なシミュレーション環境を、各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク５上に構築している。したがって、ファームウエアの開発やＣＰＵシステムの開発の環境を向上させることができ、簡単、かつ、迅速なファームウエアやＣＰＵシステムの開発を可能とすることができるとともに、オブジェクトの自由な組み合わせにより異なる入出力装置のＣＰＵシステムにおいても迅速な開発を行うことができる。



1

【特許請求の範囲】

【請求項1】シミュレーション対象となるターゲットCPUシステム用のターゲットプログラムのマシン語レベルの各命令コードをホストCPUシステム上でシミュレートして、前記ターゲットCPUシステムの動作を前記ホストCPUシステム上でシミュレートするシステムシミュレータであって、前記ターゲットプログラムを前記ホストCPUシステムで実行可能なシミュレーション環境を、前記各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク上で構築したことを特徴とするシステムシミュレータ。

【請求項2】前記システムシミュレータは、前記ターゲットCPUを模擬するCPUシミュレータと、ターゲット入出力装置を模擬するI／Oシミュレータと、前記CPUシミュレータ及びI／Oシミュレータを前記オブジェクト通信で連結するフレームワーク処理手段と、を備えていることを特徴とする請求項1記載のシステムシミュレータ。

【請求項3】前記システムシミュレータは、複数の前記ターゲットCPUシステムを模擬する複数のシステムシミュレータと、前記複数のシステムシミュレータを起動するとともに、前記複数のシステムシミュレータ相互間の関係を定義するフレームワーク処理手段と、を備えたことを特徴とする請求項1または請求項2記載のシステムシミュレータ。

【請求項4】前記ターゲットプログラムは、前記ターゲットCPUシステムのマシン語レベルの各命令コードが前記フレームワークの言語内に組み込まれ、前記システムシミュレータは、前記ターゲットプログラムの前記マシン語レベルの各命令コードを翻訳するソースコードシミュレータを備え、当該ソースコードシミュレータの翻訳した前記各命令コードを解釈して、シミュレートすることを特徴とする請求項1から請求項3のいずれかに記載のシステムシミュレータ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、システムシミュレータに関し、詳細には、フレームワーク上でシミュレーション環境を構築するシステムシミュレータに関する。

【0002】

【従来の技術】CPU（Central Processing Unit）やDSP（Digital Signal Processor）等のプロセッサを含むCPUシステム及びこのCPUシステム用のプログラムの開発においては、実際のCPUシステム（以下、ターゲットCPUシステムという。）の完成前に、当該ターゲットCPUシステム用に開発されたプログラム（以下、ターゲットプログラムという。）を、ターゲットCPUシステムとは別のCPUシステム（以下、ホストCPUシステムという。）で、ターゲットCPUシステムと模擬的に動作させるシミュレータを用いて、デバッグしたり、性能評価を行うことが行われている。

2

【0003】このようなシミュレータとしては、従来、例えば、ファイルに保存する状態情報を少なくしてもターゲットプログラムを同一の地点から再実行可能としたCPUシミュレータ（特開平8－185341号公報参照）、ターゲットプログラムのマシン語レベルの命令コードの解釈、ホストCPUの命令コードへの変換を前段階でまとめて行って、シミュレーション実行時にターゲットプログラムのマシン語レベルの命令の解釈を不要として、シミュレーションの高速化を図ったCPUシミュレーション方法及びCPUシミュレータ（特開平6－202903号公報参照）、ワークステーション上で複数のシミュレータを独立に動作させて、CPU単体の処理のシミュレーションとCPU間の通信処理のシミュレーションを同時に行うマルチプロセッサシミュレーション装置（特開平7－281925号公報参照）及び複数個のCPUの動作を並行してシミュレートするとともに各CPU相互の関係動作についてもシミュレートするシミュレーションシステム（特開平5－35534号公報参照）等が提案されている。

【0004】すなわち、上記各シミュレータは、ターゲットマシンのシミュレータのようにターゲットCPUと入出力装置をシミュレーションするものはあっても、ターゲットマシン語翻訳処理やターゲットプログラムを、別に管理しており、また、CPUシミュレータのように、ターゲットCPUシミュレータ、ターゲットマシン語翻訳処理及びターゲットプログラムを別々に管理している。

【0005】

【発明が解決しようとする課題】しかしながら、このような従来のシミュレータにあっては、ターゲットCPUシステム、ターゲットマシン語翻訳処理及びターゲットプログラムを別々に管理しているため、CPUやDSP及び入出力装置を含めたCPUシステムの開発に時間がかかるとともに、汎用性がなく、また、ターゲットCPU専用のシミュレータを開発する必要があり、迅速な開発を行う上で、改良の必要があった。

【0006】そこで、請求項1記載の発明は、ターゲットプログラムをホストCPUシステムで実行可能なシミュレーション環境を、各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク上に構築することにより、ファームウエアの開発やCPUシステムの開発の環境を向上させて、簡単、かつ、迅速なファームウエアやCPUシステムの開発を可能とするとともに、オブジェクトの自由な組み合わせにより異なる入出力装置のCPUシステムにおいても迅速な開発を行うことのできるシステムシミュレータを提供することを目的としている。

【0007】請求項2記載の発明は、システムシミュレータを、ターゲットCPUを模擬するCPUシミュレー

タと、ターゲット入出力装置を模擬するＩ／Ｏシミュレータと、ＣＰＵシミュレータ及びＩ／Ｏシミュレータをオブジェクト通信で連結するフレームワーク処理手段と、を備えたものとすることにより、入出力装置を備えたＣＰＵシステムにおいても迅速な開発を行うことのできるシステムシミュレータを提供することを目的としている。

【０００８】請求項３記載の発明は、システムシミュレータを、複数のターゲットＣＰＵシステムを模擬する複数のシステムシミュレータと、複数のシステムシミュレータを起動するとともに、複数のシステムシミュレータ相互間の関係を定義するフレームワーク処理手段と、を備えたものとすることにより、複数のシステムシミュレータを連係して動作させて適切にシミュレートし、複数のＣＰＵシステムで構成されるターゲットＣＰＵシステムを簡単、かつ、迅速に開発することのできるシステムシミュレータを提供することを目的としている。

【０００９】請求項４記載の発明は、ターゲットＣＰＵシステムのマシン語レベルの各命令コードがフレームワークの言語内に組み込まれたターゲットプログラムの当該マシン語レベルの各命令コードをソースコードシミュレータで翻訳して、シミュレートすることにより、例えば、ターゲットＣＰＵシステムのアセンブラコードをフレームワークの言語（例えば、Ｃ++言語）内に組み込んでターゲットプログラムを作成し、フレームワークの言語として、例えば、Ｃ++言語を使用する場合には、Ｃ++言語に付属しているデバッガ等で専用のシミュレータを開発する手間を省いて、シミュレータの開発を簡単かつ迅速に行うことのできるシステムシミュレータを提供することを目的としている。

【００１０】

【課題を解決するための手段】請求項１記載の発明のシステムシミュレータは、シミュレーション対象となるターゲットＣＰＵシステム用のターゲットプログラムのマシン語レベルの各命令コードをホストＣＰＵシステム上でシミュレートして、前記ターゲットＣＰＵシステムの動作を前記ホストＣＰＵシステム上でシミュレートするシステムシミュレータであって、前記ターゲットプログラムを前記ホストＣＰＵシステムで実行可能なシミュレーション環境を、前記各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク上で構築することにより、上記目的を達成している。

【００１１】上記構成によれば、ターゲットプログラムをホストＣＰＵシステムで実行可能なシミュレーション環境を、各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク上に構築しているので、ファームウエアの開発やＣＰＵシステムの開発の環境を向上させることができ、簡単、かつ、迅速なファームウエアやＣＰＵシステムの開発を可能とすることができるとともに、オブジェクトの自由な組み合わせにより異なる入出力装置のＣＰＵシステムにおいても迅速な開発を行うことができる。

【００１２】この場合、例えば、請求項２に記載するように、前記システムシミュレータは、前記ターゲットＣＰＵを模擬するＣＰＵシミュレータと、ターゲット入出力装置を模擬するＩ／Ｏシミュレータと、前記ＣＰＵシミュレータ及びＩ／Ｏシミュレータを前記オブジェクト通信で連結するフレームワーク処理手段と、を備えていてもよい。

【００１３】上記構成によれば、システムシミュレータを、ターゲットＣＰＵを模擬するＣＰＵシミュレータと、ターゲット入出力装置を模擬するＩ／Ｏシミュレータと、ＣＰＵシミュレータ及びＩ／Ｏシミュレータをオブジェクト通信で連結するフレームワーク処理手段と、を備えたものとしているので、入出力装置を備えたＣＰＵシステムにおいても迅速な開発を行うことができる。

【００１４】また、例えば、請求項３に記載するように、前記システムシミュレータは、複数の前記ターゲットＣＰＵシステムを模擬する複数のシステムシミュレータと、前記複数のシステムシミュレータを起動するとともに、前記複数のシステムシミュレータ相互間の関係を定義するフレームワーク処理手段と、を備えたものであってもよい。

【００１５】上記構成によれば、システムシミュレータを、複数のターゲットＣＰＵシステムを模擬する複数のシステムシミュレータと、複数のシステムシミュレータを起動するとともに、複数のシステムシミュレータ相互間の関係を定義するフレームワーク処理手段と、を備えたものとしているので、複数のシステムシミュレータを連係して動作させて適切にシミュレートすることができ、複数のＣＰＵシステムで構成されるターゲットＣＰＵシステムを簡単、かつ、迅速に開発することができる。

【００１６】さらに、例えば、請求項４に記載するように、前記ターゲットプログラムは、前記ターゲットＣＰＵシステムのマシン語レベルの各命令コードが前記フレームワークの言語内に組み込まれ、前記システムシミュレータは、前記ターゲットプログラムの前記マシン語レベルの各命令コードを翻訳するソースコードシミュレータを備え、当該ソースコードシミュレータの翻訳した前記各命令コードを解釈して、シミュレートするものであってもよい。

【００１７】上記構成によれば、ターゲットＣＰＵシステムのマシン語レベルの各命令コードがフレームワークの言語内に組み込まれたターゲットプログラムの当該マシン語レベルの各命令コードをソースコードシミュレータで翻訳して、シミュレートするので、例えば、ターゲットＣＰＵシステムのアセンブラコードをフレームワークの言語（例えば、Ｃ++言語）内に組み込んでターゲットプログラムを作成することができ、フレームワークの

言語として、例えば、C++言語を使用する場合には、C++言語に付属しているデバッガ等で専用のシミュレータを開発する手間を省いて、シミュレータの開発を簡単かつ迅速に行うことができる。

【００１８】

【発明の実施の形態】以下、本発明の好適な実施の形態を添付図面に基づいて詳細に説明する。なお、以下に述べる実施の形態は、本発明の好適な実施の形態であるから、技術的に好ましい種々の限定が付されているが、本発明の範囲は、以下の説明において特に本発明を限定する旨の記載がない限り、これらの態様に限られるものではない。

【００１９】図１～図４は、本発明のシステムシミュレータの第１の実施の形態を示す図であり、図１は、本発明のシステムシミュレータの第１の実施の形態を適用したシステムシミュレータ１の概念図である。

【００２０】図１において、システムシミュレータ１は、シミュレーション環境をオブジェクト指向のフレームワーク上に構築している。

【００２１】すなわち、システムシミュレータ１は、ターゲットプログラム２、ＣＰＵシミュレータ３、入出力Ｉ／Ｏシミュレータ４及びフレームワーク５等を備えており、シミュレーション対象となるターゲットＣＰＵ（Central Processing Unit ）あるいはターゲットＤＳＰ（Digital Signal Processor）（以下、総称して、ターゲットＣＰＵという。）やシミュレーション対象となる入出力装置（以下、ターゲット入出力Ｉ／Ｏという。）及びターゲット入出力Ｉ／Ｏに接続されるシミュレーション対象となる周辺機器（以下、ターゲット周辺機器）等で構成されるＣＰＵシステム（以下、ターゲットＣＰＵシステムという。）用に作成されたターゲットプログラム２（以下、単にターゲットプログラムという。）の動作を、ターゲットＣＰＵシステムとは異なるＣＰＵやＤＳＰ、入出力装置及び周辺機器等のＣＰＵシステム（以下、ホストＣＰＵシステムという。）上でシミュレートするものである。

【００２２】オブジェクト指向のシステム開発においては、一般に、ターゲットＣＰＵを模擬するコアシミュレータ（ＣＰＵシミュレータ）及びターゲットＩ／Ｏを模擬するＩ／Ｏシミュレータ、ターゲットＩ／Ｏに接続されるターゲット周辺機器を模擬するＥＸＴシミュレータ及びターゲットアセンブラソースコードを、クラスと呼ばれる単位に分割する。このクラスとは、データとそれに対する操作をまとめたモジュールであり、データは、属性と呼ばれ、操作は、メソッドと呼ばれている。

【００２３】クラスの間には、階層関係を定義することが可能であり、階層上で上位のクラスをスーパークラス、下位のクラスをサブクラスと呼ぶ。サブクラスのメソッドは、スーパークラスのメソッドを継承する。例えば、クラスＡのサブクラスとして、クラスＢがあり、クラスＡで定義したメソッドとして、メソッドａ１、ａ２があり、クラスＢで定義したメソッドに、メソッドｂ１があるとすると、クラスＢのメソッドとしては、メソッドａ１、ａ２、ｂ１になる。ここで、例えば、クラスＢでクラスＡのメソッドと同名のメソッドａ２が定義されていると、クラスＢでは、メソッドａ２は、クラスＢで定義されたものが使用される。

【００２４】そして、オブジェクト指向のフレームワークは、再利用可能なプログラムを導き出す相互稼働するクラスの集合である。

【００２５】一般に、ＣＰＵやＤＳＰは、プログラムカウンタにセットされているアドレスからプログラムを取得して、当該取得したプログラムを翻訳した後、当該プログラムの処理を実行しており、この基本動作として、外部からの要因（例えば、割り込み等）により処理内容を変えて処理を行っている。

【００２６】システムシミュレータ１は、この一連の動作をイベントの発生と受信に置き換える環境をフレームワーク５で提供することにより、行っている。

【００２７】すなわち、システムシミュレータ１では、ターゲットプログラム２、ＣＰＵシミュレータ３及び入出力Ｉ／Ｏシミュレータ４等のあらゆるオブジェクトをフレームワーク５に登録し、オブジェクト間通信を行って、ターゲットプログラム２のシミュレーションを行う。

【００２８】そのために、フレームワーク５は、ターゲットＣＰＵシステムのマシン語レベルの各命令コードをシミュレートするモジュールを抽象化したＣＰＵシミュレータ３のスーパークラス、アセンブラ翻訳のスーパークラス、抽象化した入出力シミュレータ３のスーパークラスを提供する。

【００２９】そして、フレームワーク５のスーパークラスは、広範囲なシステムシミュレーションを実現するために、オブジェクト管理とイベント発信及びイベント受信の機能を有し、システムシミュレータ１では、ハードウェアシミュレータ、ソフトウェアシミュレータ及び入出力シミュレータ等のあらゆるオブジェクトを登録する機能とオブジェクト間通信を実現できるようになっている。

【００３０】そして、ターゲットプログラム２、ＣＰＵシミュレータ３及び入出力Ｉ／Ｏシミュレータ４の各オブジェクトは、メッセージ経路６によりフレームワーク５に結ばれており、メッセージ経路６は、入出力Ｉ／Ｏシミュレータ４がシリアルポートの機能を有している機能のオブジェクトの場合、シリアルポート入出力メッセージ及びシリアルポート制御メッセージの送受信等を行う。

【００３１】システムシミュレータ１は、このオブジェクト間通信を、アプリケーション間にネットワークを構築することにより実現している。

【００３２】システムシミュレータ１は、各オブジェクトのイベントの発信・受信をフレームワーク５に登録することにより、柔軟な構成を実現している。

【００３３】このオブジェクトをフレームワーク５に登録するときの登録手続は、図２に示すように行う。すなわち、オブジェクトは、自身の宛先であるハンドル番号を取得し（ステップＳ１）、取得した自身のハンドル番号を使用して、受信可能なメッセージの種類をフレームワーク５に通知する（ステップＳ２）。次に、オブジェクトは、送信可能なメッセージの種類をフレームワーク５に通知し（ステップＳ３）、その後、起動可能なメソッドと種類をフレームワーク５に通知する（ステップＳ４）。

【００３４】一方、フレームワーク５は、メッセージの種類に応じた相手と接続する機能を有しており、オブジェクトからのメッセージの種類に応じた相手と接続して、オブジェクトの登録を受け付ける。

【００３５】すなわち、システムシミュレータ１は、ホストＣＰＵに一連の動作をイベントの発生と受信に置き換える環境をフレームワーク５上に構築して、シミュレーションを実現している。そして、ホストＣＰＵは、上述のように、その基本動作として、プログラムカウンタにセットされているアドレスからプログラムを取得し、プログラム翻訳後に実行するが、この基本動作の処理内容を、外部からの要因（割り込み等）により変えて処理する。

【００３６】システムシミュレータ１は、イベントの発生をフレームワーク５のイベント管理クラスがイベント・オブジェクトとして管理しており、このイベント発行部品は、明示的にイベント受信部品宛を記述することも可能であり、また、イベント受信部品宛を管理するイベント管理クラスで定義することも可能である。システムシミュレータ１は、このようなイベント管理手法を採用しているため、汎用的なクラスとして定義することができ、ハードウェアシミュレートを実現する場合や評価する場合に、イベントをフックするような機能を実現することができる。

【００３７】そして、システムシミュレータ１は、上述のように、フレームワーク５が、ターゲットＣＰＵシステムのマシン語レベルの各命令コードをシミュレートするモジュールを抽象化したＣＰＵシミュレータ３のスーパークラス及びアセンブラ翻訳のスーパークラスを提供するので、ターゲットプログラム２は、ターゲットＣＰＵシステムのマシン語レベルの各命令コードがフレームワークの言語内に組み込まれた状態で記述することができる。例えば、ターゲットプログラム２をC++言語で記述し、ターゲットＣＰＵシステムのマシン語レベルの各命令コードがアセンブラ言語である場合、図３に示すように、C++言語で記述されたターゲットプログラム２のｄｓｐ（）の因数に、文字列として、ターゲットＣＰＵシステムのマシン語レベルの各命令コードをアセンブラ言語で記述することにより作成されている。

【００３８】フレームワーク５は、このターゲットプログラム２のアセンブラコード文字列の翻訳処理を行ってマシン語に展開した後、ＣＰＵシミュレータ３に転送し、ＣＰＵシミュレータ３が当該命令をシミュレートする。

【００３９】ただ、マシン語レベルのジャンプ命令やサブルーチンコール命令は、厳密には、C++言語で使用されているｇｏｔｏ文や関数とは制約面で異なるが、マシン語におけるジャンプ命令は、プログラム・カウンタ（以下、必要に応じて、ＰＣという。）にセットする命令であり、ＣＡＬＬ命令は、その前に次に処理するアドレスをスタックに待避する操作を行うだけである。したがって、C++言語の文法の範囲内でアセンブラ命令記述を全て実現することは不可能であるが、アセンブラコード開発時においても、可読性を上げるために、例外的なＰＣ操作を行うようなことは、実施しないので、実際上は、問題が生じることはない。その結果、文法の制約、例えば、サブルーチン外へのジャンプを除けば、以下の手法を使用することにより、フレームワークの言語であるC++言語内にターゲットＣＰＵシステムのマシン語レベルであるアセンブラ言語の各命令コードを組み込むことができる。

【００４０】すなわち、アセンブラでは、アセンブラコードを記述する場合に、疑似命令を使用して記述するのが一般的であるが、疑似命令には、数値をラベル文字列として置き換える機能があり、ＰＣジャンプ命令及びサブルーチンコール命令においては、直接ジャンプ先のアドレスを数値で記述せずに、アドレスに対応したラベル名で記述する。

【００４１】例えば、上述のC++マクロにおいて、ジャンプ先をラベルで記述する場合には、ジャンプ先のラベルを事前に登録することにより、C++言語上のｇｏｔｏ文、あるいは、関数に置き換えることができる。

【００４２】また、アセンブラ特有の間接ジャンプ命令においては、ジャンプ先をラベルに置き換えることにより、事前に置き換えることができ、適切に処理することができる。

【００４３】そして、システムシミュレータ１は、上述のように、C++言語で記述されたターゲットプログラム２のｄｓｐ（）の因数に、文字列として、アセンブラ言語で記述されたターゲットＣＰＵシステムのマシン語レベルの各命令コードを、フレームワーク５が翻訳処理してマシン語に展開した後、ＣＰＵシミュレータ３に転送し、ＣＰＵシミュレータ３が当該命令をシミュレートする。

【００４４】例えば、図３に示した例では、図４に示すように、フレームワーク５は、ｄｓｐ（）マクロの因数文字列のアセンブラコードをマシン語に翻訳し（ステッ

プP1）、CPUシミュレータ3に当該翻訳した命令内容のシミュレートを指示する（ステップP2）。CPUシミュレータ3は、指示内容がコール命令（call命令）であるか、すなわち、サブルーチンコール命令であるかチェックし（ステップP3）、コール命令であると、関数呼び出し、すなわち、メソッドの実行を行って、処理を終了する（ステップP4）。

【0045】ステップP3で、コール命令でないときには、CPUシミュレータ3は、指示内容がジャンプ命令（jump命令）かどうかチェックし（ステップP5）、ジャンプ命令のときには、PC（プログラム・カウンタ）の値から事前に登録したラベル文字列が存在するか検索を行って、対応するジャンプ命令を実行して、処理を終了する（ステップP6）。ステップP5で、ジャンプ命令でないときには、CPUシミュレータ3は、そのまま処理を終了する。

【0046】このように、システムシミュレータ1は、ターゲットプログラム2をホストCPUシステムで実行可能なシミュレーション環境を、各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク5上に構築している。したがって、ファームウエアの開発やCPUシステムの開発の環境を向上させることができ、簡単、かつ、迅速なファームウエアやCPUシステムの開発を可能とすることができるとともに、オブジェクトの自由な組み合わせにより異なる入出力装置のCPUシステムにおいても迅速な開発を行うことができる。

【0047】また、システムシミュレータ1を、ターゲットCPUを模擬するCPUシミュレータ3と、ターゲット入出力装置を模擬する入出力I／Oシミュレータ4と、CPUシミュレータ3及び入出力I／Oシミュレータ4をオブジェクト通信で連結するフレームワーク5と、を備えたものとしているので、入出力装置を備えたCPUシステムにおいても迅速な開発を行うことができる。

【0048】さらに、システムシミュレータ1は、ターゲットCPUシステムのマシン語レベルの各命令コードがフレームワーク5の言語内に組み込まれたターゲットプログラム2の当該マシン語レベルの各命令コードをソースコードシミュレータで翻訳して、シミュレートしている。したがって、例えば、ターゲットCPUシステムのアセンブラコードをフレームワークの言語（例えば、C++言語）内に組み込んでターゲットプログラムを作成することができ、フレームワークの言語として、例えば、C++言語を使用する場合には、C++言語に付属しているデバッガ等で専用のシミュレータを開発する手間を省いて、シミュレータの開発を簡単かつ迅速に行うことができる。

【0049】図5は、発明のシステムシミュレータの第2の実施の形態を示す図であり、本実施の形態は、2つのCPUシステムに適用したシステムシミュレータである。

【0050】図5は、本発明のシステムシミュレータの第2の実施の形態を適用したシステムシミュレータ10の概念図であり、システムシミュレータ10は、システムシミュレータ20とシステムシミュレータ30がメッセージ経路40で結ばれて、ネットワークが構築された状態となっている。

【0051】システムシミュレータ20は、αターゲットプログラム21、αCPUシミュレータ22、α入出力I／Oシミュレータ23及びαフレームワーク24等を備えており、αターゲットプログラム21、αCPUシミュレータ22及びα入出力I／Oシミュレータ23の各オブジェクトは、メッセージ経路25によりαフレームワーク24に結ばれている。

【0052】システムシミュレータ30は、βターゲットプログラム31、βCPUシミュレータ32、β入出力I／Oシミュレータ33及びβフレームワーク34等を備え、βターゲットプログラム31、βCPUシミュレータ32及びβ入出力I／Oシミュレータ33の各オブジェクトは、メッセージ経路35によりβフレームワーク34に結ばれている。

【0053】そして、このαフレームワーク24とβフレームワーク34がメッセージ経路40で結ばれて、シミュレータ20とシミュレータ30がネットワークを構築した状態となっている。

【0054】各システムシミュレータ20とシステムシミュレータ30は、上記第1の実施の形態のシステムシミュレータ1と同様の機能を有しているとともに、システムシミュレータ20のフレームワーク17とシステムシミュレータ30のフレームワーク18がメッセージ経路40を介してオブジェクト通信を行う。

【0055】したがって、システムシミュレータ20とシステムシミュレータ30という2つのシステムシミュレータ20、30を連係して動作させて適切にシミュレートすることができ、複数のCPUシステムで構成されるターゲットCPUシステムを簡単、かつ、迅速に開発することができる。

【0056】なお、上記第2の実施の形態においては、2つのシステムシミュレータ20、30でシステムシミュレータ10を構築した場合について説明したが、2つのシステムシミュレータに限るものではなく、2つ以上の複数のシステムシミュレータを連係して動作させるシステムシミュレータを構築してもよい。

【0057】以上、本発明者によってなされた発明を好適な実施の形態に基づき具体的に説明したが、本発明は上記のものに限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【0058】

【発明の効果】請求項1記載の発明のシステムシミュレータによれば、ターゲットプログラムをホストＣＰＵシステムで実行可能なシミュレーション環境を、各命令コードをオブジェクト通信で授受することで、オブジェクト指向のフレームワーク上に構築しているので、ファームウエアの開発やＣＰＵシステムの開発の環境を向上させることができ、簡単、かつ、迅速なファームウエアやＣＰＵシステムの開発を可能とすることができるとともに、オブジェクトの自由な組み合わせにより異なる入出力装置のＣＰＵシステムにおいても迅速な開発を行うことができる。

【００５９】請求項2記載の発明のシステムシミュレータによれば、システムシミュレータを、ターゲットＣＰＵを模擬するＣＰＵシミュレータと、ターゲット入出力装置を模擬するＩ／Ｏシミュレータと、ＣＰＵシミュレータ及びＩ／Ｏシミュレータをオブジェクト通信で連結するフレームワーク処理手段と、を備えたものとしているので、入出力装置を備えたＣＰＵシステムにおいても迅速な開発を行うことができる。

【００６０】請求項3記載の発明のシステムシミュレータによれば、システムシミュレータを、複数のターゲットＣＰＵシステムを模擬する複数のシステムシミュレータと、複数のシステムシミュレータを起動するとともに、複数のシステムシミュレータ相互間の関係を定義するフレームワーク処理手段と、を備えたものとしているので、複数のシステムシミュレータを連係して動作させて適切にシミュレートすることができ、複数のＣＰＵシステムで構成されるターゲットＣＰＵシステムを簡単、かつ、迅速に開発することができる。

【００６１】請求項4記載の発明のシステムシミュレータによれば、ターゲットＣＰＵシステムのマシン語レベルの各命令コードがフレームワークの言語内に組み込まれたターゲットプログラムの当該マシン語レベルの各命令コードをソースコードシミュレータで翻訳して、シミュレートするので、例えば、ターゲットＣＰＵシステムのアセンブラコードをフレームワークの言語（例えば、

Ｃ++言語）内に組み込んでターゲットプログラムを作成することができ、フレームワークの言語として、例えば、Ｃ++言語を使用する場合には、Ｃ++言語に付属しているデバッガ等で専用のシミュレータを開発する手間を省いて、シミュレータの開発を簡単かつ迅速に行うことができる。

【図面の簡単な説明】

【図１】本発明のシステムシミュレータの第１の実施の形態を適用したシステムシミュレータの概念図。

【図２】図１のフレームワークにオブジェクトを登録する手順を示すフローチャート。

【図３】図１のターゲットプログラムのアッセンブラソースコードの記述例の一例を示す図。

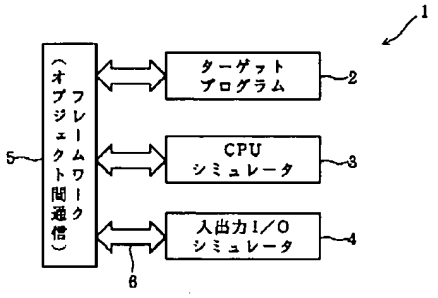
【図４】図３のターゲットプログラムの図１のシステムシミュレータでの処理手順を示すフローチャート。

【図５】本発明のシステムシミュレータの第２の実施の形態を適用したシステムシミュレータの概念図。

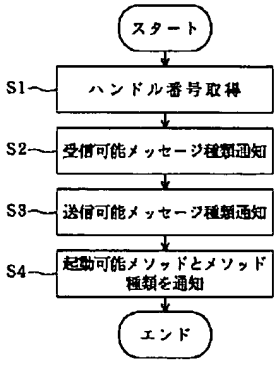
【符号の説明】

１　システムシミュレータ
２　ターゲットプログラム
３　ＣＰＵシミュレータ
４　入出力Ｉ／Ｏシミュレータ
５　フレームワーク
６　メッセージ経路
１０、２０、３０　システムシミュレータ
２１　αターゲットプログラム
２２　αＣＰＵシミュレータ
２３　α入出力Ｉ／Ｏシミュレータ
２４　αフレームワーク
２５　メッセージ経路
３１　βターゲットプログラム
３２　βＣＰＵシミュレータ
３３　β入出力Ｉ／Ｏシミュレータ
３４　βフレームワーク
３５　メッセージ経路
４０　メッセージ経路

【図１】



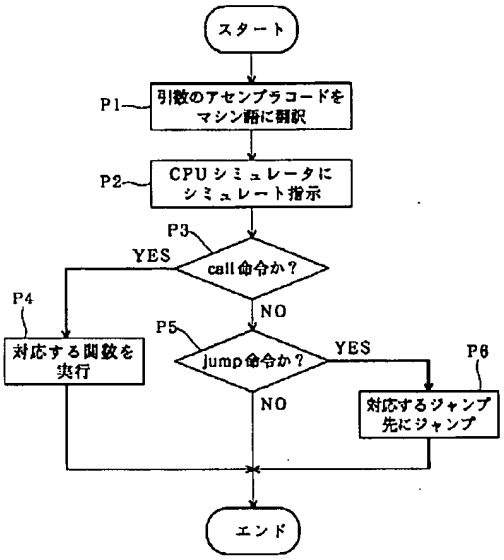
【図２】



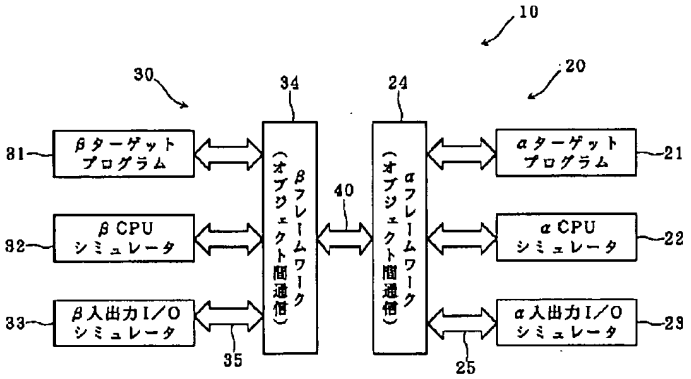
【図３】

```
void AAsm::Sub10
{
Sub1:
        dsp (   ldi   a,  Label1        );
        dsp (   jmp   a                 );
        dsp (   jmp   jmp1              );
Label1:
        dsp (   jmp   Label2            );
Label2:
}
```

【図4】


スタート
P1
引数のアセンブラコードを
マシン語に翻訳
P2
CPUシミュレータに
シミュレート指示
P3
YES
call命令か？
NO
P4
対応する関数を
実行
P5
jump命令か？
YES
P6
NO
対応するジャンプ
先にジャンプ
エンド

【図5】


10
20
30
34
24
βターゲット
プログラム
31
βフレームワーク
（オブジェクト間通信）
αフレームワーク
（オブジェクト間通信）
αターゲット
プログラム
21
40
β CPU
シミュレータ
32
α CPU
シミュレータ
22
β入出力I/O
シミュレータ
33
35
25
α入出力I/O
シミュレータ
23